DATABASE THERAPY: A practitioner's experiences

F. Alfredo Rego

Rego Software Pty
Calle del Arco 24
Antigua, GUATEMALA

Telephone (502-2) 324336
Telex 4192 TELTRO GU

ABSTRACT

I will describe my first-hand experiences on some fundamental aspects of database therapy:

Prevention: A bit of prevention can save many megabytes worth of reloading.

Periodic checkups: There are procedures to test, diagnose and report database errors and faults. And your own users may very well be your best detectors.

Treatment: There are good procedures for the correction of database errors and faults. They are still primitive but can be very effective in those cases they can now handle. And they are impressively learning to deal with new cases!

Follow-up. You cannot afford to lower your guard. You must follow up. Always.

INTRODUCTION

I have had the privilege to visit hundreds of HP3000 computer installations in all continents and in many islands. There are "radical" differences in applications, people, software, hardware, and environment. But all of these HP3000 computer installations have the very same purpose: To maintain a bunch of bits. The specific patterns formed by their pet configurations of such zillions of bits are as varied as they can be, but our fellow HP3000 users all over the world are doing exactly the same thing. It is very important to remember this!

A bunch of bits: That is all that we really have in any computer system in general and in any database in particular. A bunch of bits, some which are supposed to be ON and some which are supposed to be OFF.

To be ON, or not to be ON: that is the BIT question!

Who is authorized to decide which bits are supposed to be on or off? Who is responsible for detecting flip-flopped bits? Who is able to correct runaway bits? Let's share some thoughts and feelings on these challenging subjects.

PREVENTION

A bit of prevention can save many megabytes worth of reloading. My own personal bias favors PREVENTION ABOVE ANYTHING ELSE. I believe prevention has the greatest possible payoff, especially when the stakes are high.

Databases face serious health hazards. Let's discuss a few, beginning with the more innocent-looking ones. And let's see what preventive action you can take.

Sloppy and/or disgruntled people are the worst possible hazard to a database and to the whole computer system. If you cannot keep these jewels under control, you might as well forget everything that follows.

People can (and do) misuse software. They can use QUERY to find all the entries that meet certain criteria and then delete them. They can accomplish the same (good? bad?) thing with a ten-line BASIC program or with the equivalent 100-line structured-COBOL program. It does not matter HOW they do it. If those entries are supposed to be there (according to you), then YOU should take some preventive action. For instance:

- Use IMAGE LOGGING and Bob Green's DBAUDIT (see reference 1) to find out who did what, when, to which entries.

- Use DBUTIL to disable QUERY-B write access. Or assign a password to QUERY. Or remove QUERY from your system.

- Take advantage of the powerful password-security mechanism provided by IMAGE to restrict write-access to sensitive items or sets. Change these passwords every now and then. Be sure to write them down and store them in a cool, dry place which is NOT accessible to your enemies and/or friends.

- Assign a maintenance word to every database with DBUTIL (at creation time, or later on). Change this maintenance word periodically. Assign a password to DBUTIL itself, so people will have a harder time if/when they decide to ERASE or PURGE your databases.

Innocent-looking application programs, as you well know, are one of the worst threats to the consistency of a database. Your best strategy is to build a set of application programs that spend their lives checking the application-dependent consistency of your database. You will have to face some painful decisions regarding tradeoffs here. But something and somebody must check things occasionally just to be sure that obvious errors have not crept in.

Remember that your own applications software is responsible for keeping the semantic consistency of your database. Nobody else except you and your staff knows anything about your specific design and implementation. Therefore, _you_ must make sure that the effect of your applications software is constantly monitored by your end-users, by your quality-control and auditing staff, and (last but not least) by your programmers.

Remember also that a database is an integrated entity that has two distinct elements: the BASE system (IMAGE in our case) and the DATA that lives there. The best database management system in the world will be useless if the data you keep in it has no meaning. And it is the main responsibility of all your application programs to maintain the validity and usefulness of that data!

People can use ACCOUNT MANAGER capabilities to purge the database's group. Even worse, they can use SYSTEM MANAGER capabilities to purge the database's account. You can run LISTDIR2.PUB.SYS to list all the accounts and all the users in your system, to check their capabilities. To protect yourself, stand right at the line printer when the printout comes out. Otherwise, somebody will see it and everybody will know everybody's capabilities! Be sure that no one has more MPE capabilities than the strict minimum necessary to operate.

Privileged users (and the system operator) can use SYSDUMP, STORE and RESTORE with IMAGE databases. As long as they know what they are doing and as long as they store/restore fully-consistent collections of privileged IMAGE MPE files, everything will be all right. But the moment they make one mistake, NOTHING will be right! This is why the modules DBSTORE and DBRESTOR were designed by Hewlett-Packard specifically to store/restore IMAGE databases. They do some reasonableness checks to increase the probability that you are backing up or restoring a consistent database. And they mark the database on disc indicating the backup date and time (an important thing for logging and recovery purposes). ADAGER's BACKUP module is functionally equivalent to DBSTORE and DBRESTOR, but it uses just a fraction of their time and tape resources. As an added preventive measure, ADAGER encrypts the root file as it backs it up to tape so that nobody can FCOPY it to the line printer to find out all the user-class passwords. (MORAL: lock up your database backup tapes just as if they were cash. Never forget that your database may be much more valuable than cash!)

People can store things on the wrong set of tapes, clobbering whatever data was on the tapes! And people can restore from the wrong set of tapes, clobbering whatever data was on disc! A well-organized tape-library system is a good investment, especially if it is itself computerized (after all, you are trying, precisely, to protect yourself from sloppy operators...)

People can physically keep your backup tapes in a hostile environment, thereby rendering them useless. I recommend professional handling of your off-site tape storage. And I recommend that you periodically check the validity of the tapes kept both on-site and off-site. What would happen if you had to restore some file from some tape that was physically impossible to read?

People can fail to backup the system at all. Whether they do it intentionally or innocently is irrelevant: the catastrophic consequences are exactly the same. How do you know that the tapes that are supposed to contain your full sysdumps really contain them? Backing things up is a chore. How do you know that your people are not taking shortcuts? I know a user who has an HP3000 machine dedicated 8 hours a day to just a single purpose: a RELOAD from the sysdump tapes produced by other computers. If any reload has any difficulties whatsoever, he takes immediate action to correct the problem while it is IMPORTANT but not URGENT. Most people wait until a problem is IMPORTANT, URGENT, and IMPOSSIBLE to solve within the given time/resource constraints. This user was one of these people. After a near-catastrophe, he learned that any investment in prevention pays handsome dividends in terms of everyday health. Both his and his database's.

Please keep in mind that computerized databases are not exempt, by any means, from ENTROPY: Any system degrades to an ultimate state of inert uniformity. But please also keep in mind that you can delay your database's inevitable failure and decline. You can keep your database in a good state of repair, efficiency, validity and effectiveness. But you must be willing to invest in PREVENTIVE MAINTENANCE. Otherwise, you and your database are doomed.

Your problem, as a manager (of the whole universe, of a country, of a company, of a department, of a computer system, of a program, of even one bit), is always this: You must first choose, out of an unlimited collection of possible objectives, the ONE goal that you want to reach; and then you must also choose, out of a very limited collection of resources, those few resources that will help you reach your goal in a finite time.

These choices are extremely difficult. There is no question about that! But you can make your life easier if you decrease your collection of possible goals, if you increase your collection of resources, or if you combine both approaches.

In the specific case of your HP3000 computer system, you are very fortunate. There are many good people (in the Hewlett-Packard Company, in the HP3000 vendor community, in the HP3000 user community and in the directors and staff of the Users Group). These people have done, are doing and will continue to do excellent work, whose end result means that you can concentrate a large number of extended resources on YOUR objectives. There are many "possible goals" that you do NOT have to worry about unless you really want to. If you want an operating system, a database management system, a report writer, a word processor, an editor, etc., SOMEBODY else has already spent endless hours and lots of valuable resources to make these things available to you. Likewise, if you want some warnings about things that you should treat with care and respect, somebody else has spent endless hours and lots of valuable resources and has published these warnings for everybody's benefit.

Two recent examples are Gerald W. Davison's article on IMAGE LOCKING and Application Design (see reference 2) and my article on Design and Maintenance Criteria for IMAGE Databases (see reference 3). If you want an extended bibliography and a list of reference materials, please write to me and I will be happy to send you a printout of my latest computerized information. Rick Bergquist, the author of DBLOADNG and the User's Group Interface Committe member for IMAGE would also like to hear from you (see reference 4).

A computer (HP3000 or otherwise) is just as good as the electrical power that feeds it. My company has only a small Series 30 (Koala) but we have cheerfully invested in an uninterruptible power supply that costs as much as one disc drive. I personally authorized that procurement since I firmly believe in the value of PREVENTION. Instead of having one more disc drive, I prefer to preserve whatever we have! I will not accept growth at the expense of reliability.

A computer room is NOT a social club. Keep traffic to a minimum. Do not use it to store things which may be harmful to your equipment. Make your operators strictly accountable for everything that moves (and does not move) in the computer room, just as if they were bank tellers. After all, your information may be easily worth millions of Krugerrands.

Most systems software is amazingly water-tight. Systems software people tend to be careful to the point of being paranoid. And the good systems software available for the HP3000 is truly outstanding. Just check the DATAPRO ratings for Hewlett-Packards systems like IMAGE and for independent systems like QEDIT and ADAGER or talk to the person sitting next to you. What is not so water-tight is the sloppy or malicious use of systems software.

Take QEDIT as an example. If somebody deletes a few lines from some of your source programs (especially lines that contain "IF" statements), your whole computer system may produce strange results. Nevertheless, QEDIT is completely innocent. You must develop procedures to decrease the probability of this unpleasant happening. You may want to use Larry Simonsen's program to compare source files (see reference 5). And you may want to implement a computerized system to keep track of changes to your source files. We have IMAGE logging. We should also have SOURCE-FILE logging.

Take ADAGER as another example. If somebody adds a field to a data set or reshuffles the fields in a data set without recompiling ALL affected application programs to update their buffer definitions, your whole computer system may produce some unusual results also. Nevertheless, ADAGER is totally innocent. You must also develop procedures to decrease the probability of this occurrence. You may want to use ADAGER's SCHEMA and XRAY functions to get periodic snapshots of your database and to compare them with your reference documents. Logging ADAGER database changes will help you in this matter. Anyhow, before diving into the pool, your applications program should check whether the pool has any water in it or not! A simple call to DBINFO, for instance, can easily tell your program that a change in a data set's entry definition has taken place. If this is the case and if your program is not qualified to handle the non-standard situation, it should report the discrepancies and quit before it clobbers anything.

Take DBUNLOAD/DBLOAD as still another example. Both Hewlett-Packard programs handle full entries (see reference 6). If you delete or add fields, if you redefine items, or if you delete or add sets, the fact that the unload/load cycle completes successfully does not mean a thing: your database may still be easily clobbered anyway!

One of the best preventive measures you can take is to establish a standard DATA DIRECTORY & DICTIONARY system. Tipton Cole is the Users Group Committee Chairman for this project. He would like to hear from you if you are interested in these ideas (see reference 7).

Logging and recovery are worthy things in themselves. But you can never be protected 100%. For instance, if a system failure happens in the middle of a logging/recovery cycle, your database may be left in an inconsistent state. Rick Bergquist has done extensive work in this area and you may want to contact him for further references (see reference 4). For the time being, HewlettPackard's recovery system allows only roll-forward. Rick's method allows also backout recovery.

The Users Group Contributed Library is certainly full of very valuable software (getting close to two million U.S. Dollars in value). But you must check its programs carefully to be sure you understand them. And you must maintain these programs to keep them in step with the times (and the operating system releases!) For instance, if you installed OVERLORD or SOO a couple of years ago, you will have to change them when you install MPE IV. The Central Contributed Library Office will send you the Contributed Library Tape with the appropriate software but YOU still have to install it yourself. If you have any questions whatsoever, contact Mark Wysoski, the most knowledgeable person regarding the Contributed Library. He will help you and he will guide you, whether you need to use a program or contribute a program. He usually knows whether any bugs are outstanding on a given program that may cause problems in your site. And all the other members of the group would appreciate it if you would report any peculiarities that you observe in the contributed programs. This community-wide network is essential for all of us. Please help. (See reference 8).

Even though Hewlett-Packard equipment is very reliable, it may fail, since it is not exempt from ENTROPY either. The obvious things like printers do not worry me too much. But the subtle things like CPU chips do worry me. My Customer Engineer in Guatemala, Johnny Siemon, taught me how to run the CPU diagnostics. I run them once a day (you can never run too many diagnostics!) If any test whatsoever fails, I do a few things like resetting the CPU and then giving the test another try. If it fails again, I pull the plug and call him immediately, with the exact chip number. He arrives with the right parts and I am up and running very soon. I would much rather do this bit of HARDWARE PREVENTIVE work myself than risk the possibility of "unusual" address calculations on disc for file-write operations, for instance. (Just for the record: I have only found one bad chip in more than a year of operation. Johnny had the part shipped from Guatemala City to Antigua and installed while I had lunch. So, for all practical purposes, our HP3000 has never been down. This is the best motivation we have to keep our budget for investments in preventive measures UP. At any time. All the time. At any cost. It turns out to be dirt-cheap in the long term. And our strategy, as a company, is quite biased towards the long term by choice).

I am amazed at the kinds of cruel environments in which the HP3000 computer survives. Nevertheless, there are a few things that you should watch out for. Controllable by you (at a cost) are things like smoke, dust, and electrical power. I would suggest that any investment you make to control these parameters will pay for itself many times over. Uncontrollable parameters like cosmic rays, earthquakes, floods, wars, etc., will not affect you as much if you have previously established a consistent program of backup and recovery, complete with emergency drills to see how your people behave under pressure. There is a large amount of literature on this subject and I will be happy to mail you some references if you wish. I would also like to hear from you if you have any suggestions or comments regarding the ways that you have implemented in your site.

PERIODIC CHECKUPS

Webster defines DIAGNOSIS as the art of identifying a disease from its signs and symptoms, as an investigation or analysis of the cause or nature of a condition, situation or problem.

I have some friends who can smell a rat in no time at all. I have other friends who could easily spend a year testing a system, full time, without ever finding the errors that do exist in the system. I have decided that the degree of "education", "training", "certification", etc. associated with successful diagnosticians seems to be totally irrelevant. A certain amount of raw gut feeling is in effect here, which defies easy rules and pigeon-holing. Just like music, poetry, painting, photography, architecture and other creative endeavors, "some people have it and some do not". I have simply decided that I prefer to invest my time IDENTIFYING good people and then relying on their judgment rather than to waste my time trying to train untrainable people. This works out to be to everybody's advantage (there is nothing more pitiful than seeing somebody who has NO aptitude whatsoever trying to cope with concepts or actions which are dramatically beyond that somebody's grasp).

You may have noticed that some of the most "naive" users of your computer system can detect, immediately, "when, how, and why" something is wrong. Do NOT underestimate them! Usually they are the very clerks who know "their" data and information quite intimately. Be sure to include such users in your early-warning system. They may be much more valuable to you than some sophisticated programmers who spend all of your computer's resources navigating through masses of data and sailing right past the most obvious problems which, if not corrected immediately, will cause enormous losses. Do not forget that education, salary, title, position, age, sex, etc., are utterly irrelevant. If the Vicepresident in charge of Information Systems and the payroll clerk do not agree on something, I would always place my bet on the clerk's side (Not on matters of policy but on matters of DETECTING ERRORS, of course!)

"Errors" are a hard thing to define. We will use the term here in a somewhat subjective manner related to our "desirability" of certain things. We will say that an ERRONEOUS SYSTEM, given a set of TRIGGERING CONDITIONS, will cause a set of UNDESIRABLE EFFECTS. You certainly do NOT want undesirable effects under any circumstances or conditions whatsoever. Unfortunately, this is an impossible goal, since you could not even dream up ALL possible sets of conditions, much less test and certify them!

A good DIAGNOSIS SYSTEM should analyze the behavior of your system under a meaningful subset of the total CONDITIONS SPACE. This is easier said than done. Defining this meaningful subset is a very difficult task. Even more, the fact that a diagnosis system does not uncover undesirable effects under its subspace of conditions does NOT mean at all that you have a correct system. You may be just lucky. Your set of tested conditions may be such that those portions of your system which produce undesirable effects are just not exercised. But those nasty portions of your system are <u>precisely</u> the ones you want to discover because they lurk there, ready to jump whenever their set of conditions happens (according to Murphy, this will happen at the worst possible time).

A good system of diagnosis goes much beyond just saying "no errors detected". It devises as many sets of linearly-independent conditions as possible to see how the system reacts. If it detects undesirable effects, it issues appropriate messages saying "the system fails with this set of conditions (and describes them)", "the effects are (and describes them)", "the error(s) probably reside(s) in such and such part(s) of the system". Instead of just telling you that something is wrong, it tells you how precisely it is wrong, perhaps why it got that way so you may avoid it next time, and how you can fix it.

Please notice that a good diagnosis system must be nasty and sadistic by nature. It has as its primary objective to FIND ERRORS, <u>not</u> to certify a system as being error-free (there is no such system anyway!) A good diagnosis system must also be extremely patient and humble, since it will fail many times. Please keep in mind that there is a psychological inversion in effect here: A good diagnosis system fails if it does not detect any errors. And most of the time it will not detect any errors, since we hope and assume that the entity being tested is reasonably error-free. Naturally, if you are testing a "lemon", almost any diagnosis system will have a successful run. But that is the trivial case!

It is unrealistic to even attempt to test ALL POSSIBLE sets of conditions. It is unrealistic to even suggest to test the much smaller subset of ALL PROBABLE sets of conditions. The challenge for the diagnosis system then is to find the MAXIMUM number of errors and faults with the MINIMUM investment in diagnosis tools and test runs.

A large number of diagnosis tools and diagnostic runs may be totally worthless if they are poorly designed and implemented (they will simply say "end of step 154, no errors found" or "error 538 found in step 47, loop 10457"). On the other hand, a small number of these tools, intelligently put together, may be very effective in finding elusive errors. Such a system, naturally, is very difficult to design and implement. We are just now in the threshold of a new era in software engineering.

It turns out that Zeno's paradox applies equally well to finding errors as it does to shooting arrows: Finding the first "half" of all errors present in the system requires some effort; finding half of the remaining errors takes about the same amount of effort, and so on ad infinitum, with the last difficult bugs taking more time to find than the first (to quote from Brooks' "The Mythical Man Month"). We have here something akin to a half-life type of situation for bugs!

For specific programs that you can run now to diagnose IMAGE database errors, please write to me and I will mail you an up-to-the-minute list of references. In the meantime, here is a partial list:

- DBCHECK (also called STRUKCHK). In the Contributed Library. Distributed also in the QEDIT and ADAGER tapes. It cruises through either one or all data sets in a database. It does a fairly thorough examination of all the structural parts of the data sets and reports, in great octal detail, those things that it does not like. A few of those things may not be errors at all but it always nice to know that the program detected them.

- DBLOADNG. In the Contributed Library, QEDIT and ADAGER tapes also. It also cruises through a database's data sets and produces a summary report on the general state of the database. It indicates conditions that may require attention (like master capacities which are not prime numbers, or master data sets which are more than 80% full, etc.)

- ADAGER's XRAY and SCHEMA functions, as well as QUERY's FORM command, will give you a fairly good idea of what is really in your database (as opposed to what you THINK is in your database!)

- DBSTORE/DBRESTOR (or ADAGER's BACKUP function) will do some reasonableness checking to make sure that, at least, you have all the MPE privileged files that your database is supposed to have.

TREATMENT

There are good procedures for the detection and correction of database errors and faults. They are still primitive but can be very effective in those cases they can now handle. And they are impressively learning to deal with new cases.

Hewlett-Packard has developed a program which is a Systems-Engineer tool. Please attend Duane Souder's session ("DBTEST: A database structure test and repair facility") and please read his paper in these Orlando-81 proceedings.

Rego Software PTY has developed an ADAGER function called NURSE which is currently undergoing pre-release tests. Many ADAGER modules, as a matter of fact, have already incorporated some of NURSE's functions and have been released since 1980. NURSE has been developed with the help of many ADAGER users all over the world. Whether you are an ADAGER user or not, I would appreciate your help during our ongoing research and development efforts. Please provide us with samples of database errors so we can analyze them and classify them.

ADAGER's NURSE function is developing in an evolutionary way. It is now a recognized expert on those categories of database errors that it has catalogued in its "vocabulary". And it provides a commented report on those database errors that it has not (yet) learned to correct. The ADAGER-formatted file created by NURSE contains enough information to allow us to incorporate a new linearly-independent vector in NURSE's vector-space base. This way, the cycle keeps on going: Our friends (and NURSE) report to us new categories; we study them and incorporate them into NURSE's bag of tricks... and so on!

Wayne Benoit's DBGROOM (in the Contributed library) allows you to directly manipulate structural elements in an IMAGE database. It allows you better access than if you were using DISKEDIT. But you are still fully responsible for all that octal thinking! If you blow even one vital root-file bit, nobody will warn you. This is a great tool if you REALLY know what you are doing (and Wayne Benoit certainly knows what <u>HE</u> is doing, since his knowledge of the internals of IMAGE is very thorough); but this tool can destroy everything if you do not know what you are doing.

Hewlett-Packard's PRIVILEGED DEBUG and DISKEDIT allow you unlimited access to every single bit in memory and on disc. You can fix ANYTHING, in theory, by means of these tools. But you have to think strictly in terms of bits. Zillions of them. Each with a vital meaning. I sincerely do not believe that anybody can use these tools for reasonably complicated surgical operations on databases. But they are certainly available anyway to anybody authorized by your HP3000 System Manager.

FOLLOW-UP

You cannot afford to lower your guard. You must follow up. Always. One way to follow up is to keep up with IMAGE-related research and development. Read the Users Group Journals and Newsletters. Attend your local Users Groups meetings. Call your HP3000 colleagues whenever you have any questions, comments or suggestions. Contact your Hewlett-Packard Systems Engineer and your independent software suppliers.

Hewlett-Packard's IMAGE is an excellent database management system. It is simple. It is reliable. It is robust. Hewlett-Packard is committed to its continued improvement. Rego Software PTY is committed to ADAGER's continued improvement. Other software vendors are committed to the continued improvement of their IMAGE-related products.

If all of us in the world-wide Hewlett-Packard family integrate the use of our combined resources for the attainment of our common goal, we can all look forward to IMAGE's unbounded success. THIS IS OUR COMMON GOAL. Let us all follow up!

Thank you.

REFERENCES

1 - Robert M. Green (604) 943-8021 Telex 04-352848
Robelle Consulting Ltd.
5421 - 10th Avenue, Suite 130
Delta, British Columbia V4M 3T9
Canada

2 - IMAGE LOCKING AND APPLICATION DESIGN
Hewlett-Packard Users Group Journal,
First Quarter 1981 Vol IV No. 1
Gerald W. Davison
Argonne National Laboratory
9700 South Cass Avenue
Argonne, Illinois 60439
U.S.A.

3 - DESIGN AND MAINTENANCE CRITERIA FOR IMAGE/3000
Hewlett-Packard Users Group Journal,
Fourth Quarter 1980 Vol III No. 4
F. Alfredo Rego
Rego Software PTY
Calle del Arco 24
Antigua
Guatemala

4 - OPTIMIZING IMAGE: AN INTRODUCTION
Hewlett-Packard Users Group Journal,
Second Quarter 1980 Vol III No. 2
Rick Bergquist (415) 573-9481
American Management Systems
561 Pilgrim Drive
San Mateo, California 94404
U.S.A.

5 - Larry Simonsen (801) 489-8611
VALTEK
Mountain Springs Parkway
Springville, Utah 84663
U.S.A.

6 - IMAGE/3000 REFERENCE MANUAL
Hewlett-Packard Company (408) 725-8111
19447 Pruneridge Avenue
Cupertino, California 95014
U.S.A.

7 - Tipton Cole (512) 452-0247
Cole & Van Sickle
7701 Cameron Road
Austin, Texas 78761
U.S.A.

8 - Mark C. Wysoski (509) 527-5360
Manager, Hewlett-Packard Users Group
Contributed Library
Whitman College
Walla Walla, Washington 99362
U.S.A.